Homework 1a

February 19, 2019

1 Problem 1

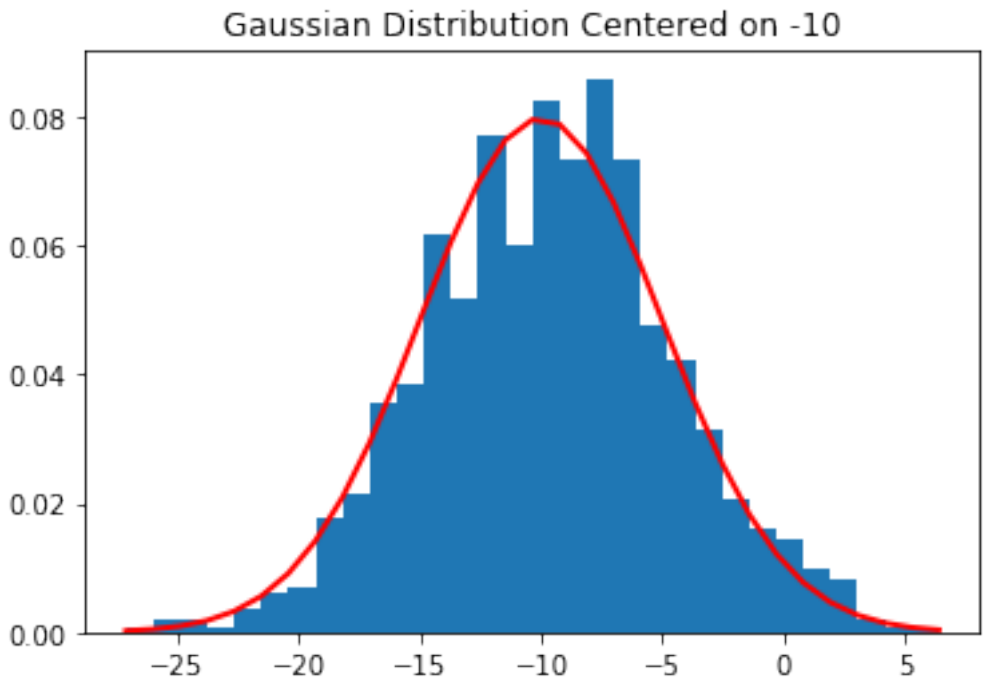
In [2]:
```
import numpy as np

p1_mu_neg = np.random.normal(-10, 5, 1000)
p1_mu_positive = np.random.normal(10, 5, 1000)
```

1.1 (a)

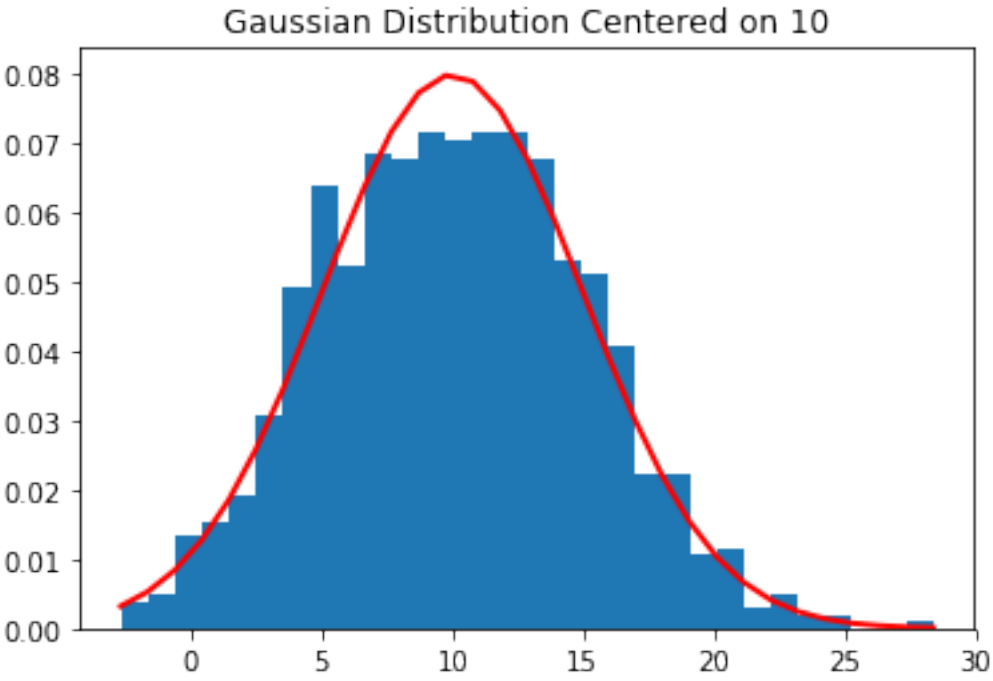
In [3]:
```
import matplotlib.pyplot as plt
def plot_gauss_distribution(distribution, title, mu, sigma):
    count, bins, ignored = plt.hist(distribution, 30, normed=True)
    plt.title(title)
    plt.plot(bins, 1/(sigma * np.sqrt(2 * np.pi)) *
                   np.exp( - (bins - mu)**2 / (2 * sigma**2) ),
             linewidth=2, color='r')
    plt.show()
#     plt.clf()
```

In [4]:
```
plot_gauss_distribution(p1_mu_neg, 'Gaussian Distribution Centered on -10', -10, 5)
```

```
/usr/local/lib/python3.7/site-packages/matplotlib/axes/_axes.py:6521: MatplotlibDeprecationWarn
The 'normed' kwarg was deprecated in Matplotlib 2.1 and will be removed in 3.1. Use 'density' i
  alternative="'density'", removal="3.1")
```



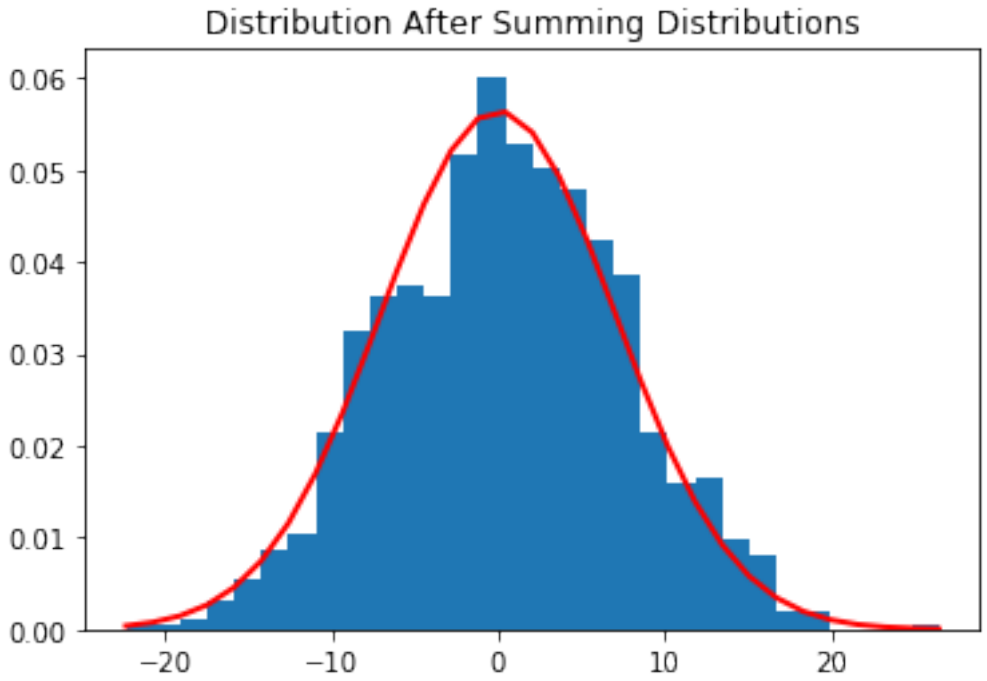
In [5]: plot_gauss_distribution(p1_mu_positive, "Gaussian Distribution Centered on 10", 10, 5)



1.1.1 1(a) Answer

By looking at the plot below, we see that the new distribution is centered at 0 with with a higher variance than the first two.

```
In [6]: p1_new_sum = [p1_mu_neg[i] + p1_mu_positive[i] for i in range(len(p1_mu_neg))]
        plot_gauss_distribution(p1_new_sum, "Distribution After Summing Distributions", 0, 7.0
```



1.2 (b) Answer

My estimation for the mean is 0 and the variance is approximately 50.

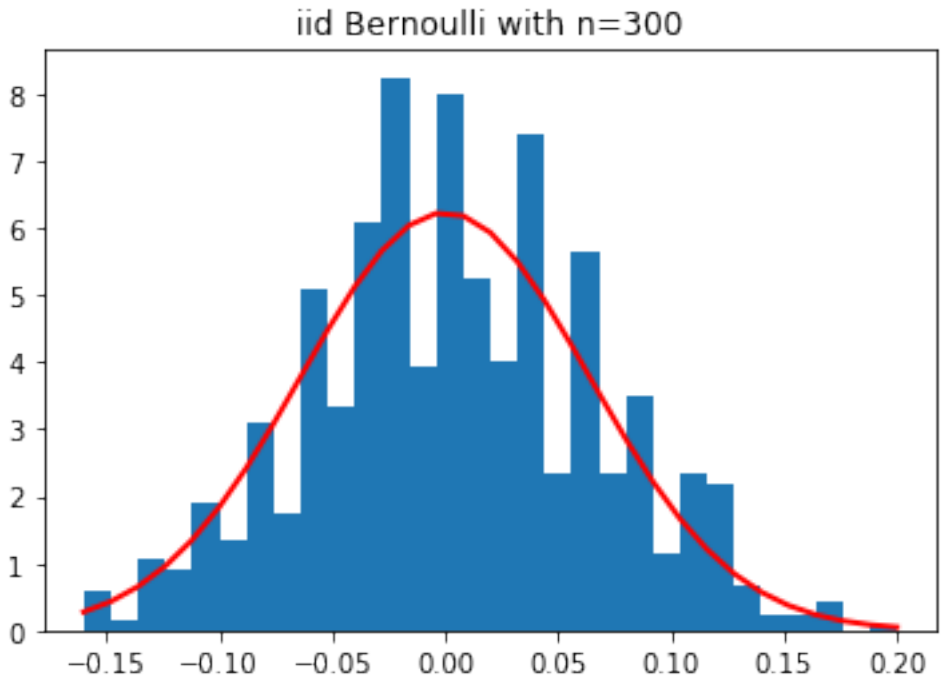
2 Problem 2

```
In [7]: import random
        import statistics as stat

        def generate_random_bernoulli():
            return 1 if random.randint(0, 1) else -1

        def generate_z(n):
            return 1 / n * sum([generate_random_bernoulli() for _ in range(n)])

        def generate_1000_z(n):
            return [generate_z(n) for _ in range(1000)]
```

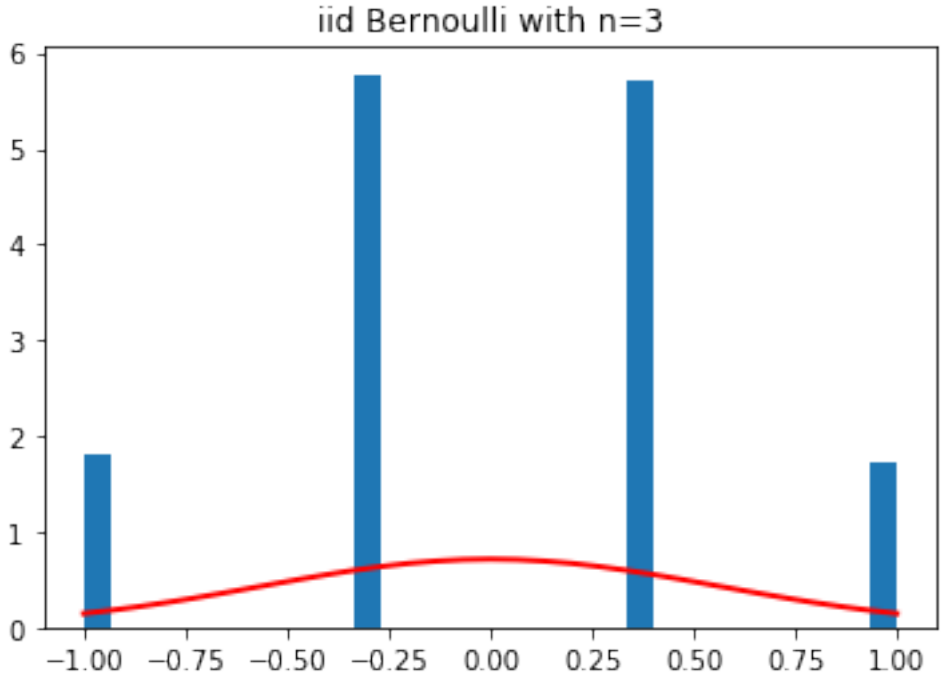
```
p2_z3 = generate_1000_z(3)
p2_z5 = generate_1000_z(5)
p2_z8 = generate_1000_z(8)
p2_z50 = generate_1000_z(50)
p2_z300 = generate_1000_z(250)

print("n=3")
print(stat.stdev(p2_z3))
print(stat.mean(p2_z3))
print("n=8")
print(stat.stdev(p2_z8))
print("n=300")
print(stat.stdev(p2_z300))
print(stat.mean(p2_z300))
```

```
n=3
0.5651342407281449
-0.007333333333333333
n=8
0.34306359487729116
n=300
0.06371593414730937
0.0036160000000000003
```

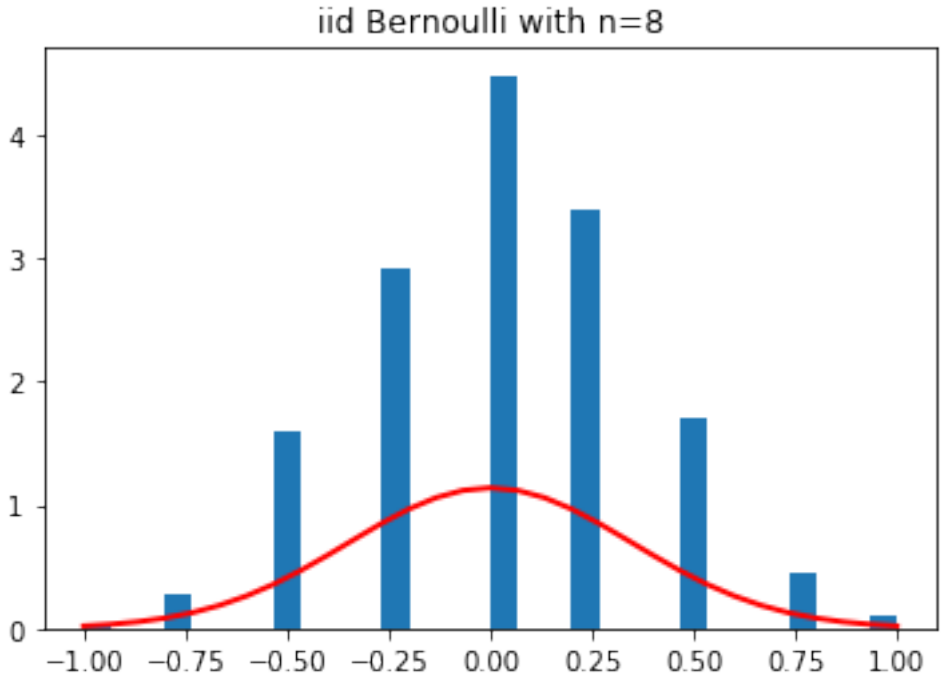
In [8]:
```
plot_gauss_distribution(p2_z300, "iid Bernoulli with n=300", 0, .064)
```



In [9]: plot_gauss_distribution(p2_z3, "iid Bernoulli with n=3", 0, .56)



In [10]: plot_gauss_distribution(p2_z8, "iid Bernoulli with n=8", 0, .35)



3 Problem 3

```
In [11]: p3_sample = np.random.normal(0, 5, 25000)

In [12]: p3_avg = sum(p3_sample) / len(p3_sample)
         p3_avg

Out[12]: -0.02458546249948946

In [13]: import math

         p3_std_dev = math.sqrt(sum([(x-p3_avg) ** 2 for x in p3_sample]) / (25000-1))
         p3_std_dev

Out[13]: 4.975409293621266
```

3.0.1 Problem 3 Answer

I estimaged the average to be -.025 and the standard deviation to be 4.975.

4 Problem 4

```
In [14]: p4_dist = np.random.multivariate_normal([-5,5], [[20,.8],[.8,30]],10000)

In [15]: from functools import reduce

         p4_x_mean = reduce(lambda y, z: y+z, [t[0] for t in p4_dist]) / len(p4_dist)
         p4_y_mean = reduce(lambda x,z: z+x, [t[1] for t in p4_dist]) / len(p4_dist)

In [16]: p4_x_mean

Out[16]: -5.050979798046953

In [17]: p4_y_mean

Out[17]: 5.001967553365301

In [18]: p4_variance_matrix =[
             [1/len(p4_dist - 1) * sum((t[0]-p4_x_mean)**2 for t in p4_dist),
              1/len(p4_dist - 1) * sum((t[0] - p4_x_mean) * (t[1] - p4_y_mean) for t in p4_dist
             [1/len(p4_dist - 1) * sum((t[0] - p4_x_mean) * (t[1] - p4_y_mean) for t in p4_dist
              1/len(p4_dist - 1) * sum((t[1] - p4_y_mean)**2 for t in p4_dist)]]

In [19]: p4_variance_matrix

Out[19]: [[19.701755994633658, 0.7885865322933491],
          [0.7885865322933491, 28.87312624223446]]
```

5 Problem 5

```
In [20]: import pandas as pd

         df = pd.read_csv('PatientData.csv', header=None)
         df
```

Out[20]:

	0	1	2	3	4	5	6	7	8	9	...	270	271	272	\
0	75	0	190	80	91	193	371	174	121	-16	...	0.0	9.0	-0.9	
1	56	1	165	64	81	174	401	149	39	25	...	0.0	8.5	0.0	
2	54	0	172	95	138	163	386	185	102	96	...	0.0	9.5	-2.4	
3	55	0	175	94	100	202	380	179	143	28	...	0.0	12.2	-2.2	
4	75	0	190	80	88	181	360	177	103	-16	...	0.0	13.1	-3.6	
5	13	0	169	51	100	167	321	174	91	107	...	-0.6	12.2	-2.8	
6	40	1	160	52	77	129	377	133	77	77	...	0.0	6.5	0.0	
7	49	1	162	54	78	0	376	157	70	67	...	0.0	8.2	-1.9	
8	44	0	168	56	84	118	354	160	63	61	...	0.0	7.0	-1.3	
9	50	1	167	67	89	130	383	156	73	85	...	-0.6	10.8	-1.7	
10	62	0	170	72	102	135	401	156	83	72	...	-0.5	9.0	-2.0	
11	45	1	165	86	77	143	373	150	65	12	...	0.0	4.4	-2.2	
12	54	1	172	58	78	155	382	163	81	-24	...	0.0	6.3	-2.1	
13	30	0	170	73	91	180	355	157	104	68	...	-0.9	12.3	0.0	
14	44	1	160	88	77	158	399	163	94	46	...	-0.6	12.4	0.0	
15	47	1	150	48	75	132	350	169	65	36	...	0.0	7.7	-0.8	
16	47	0	171	59	82	145	347	169	61	77	...	0.0	9.4	-1.7	
17	46	1	158	58	70	120	353	122	52	57	...	0.0	6.6	0.0	
18	73	0	165	63	91	154	392	175	83	73	...	0.0	5.7	0.0	
19	57	1	166	72	82	181	399	158	79	-12	...	0.0	7.7	-0.9	
20	28	1	160	58	83	251	383	189	183	50	...	-0.6	9.1	-1.4	
21	45	0	169	67	90	122	336	177	78	81	...	-0.6	8.3	-1.8	
22	36	1	153	75	71	132	364	169	82	62	...	0.0	8.9	-1.0	
23	57	1	165	59	75	157	406	143	92	4	...	0.0	6.7	-0.5	
24	40	1	153	55	82	140	388	149	82	52	...	0.0	13.6	0.0	
25	44	0	169	80	109	128	382	195	60	-34	...	0.0	6.9	0.0	
26	34	0	170	73	94	186	373	224	125	90	...	0.0	15.3	-1.1	
27	31	1	160	54	95	161	407	168	83	10	...	0.0	12.7	-1.8	
28	56	1	164	65	90	164	420	381	99	-8	...	0.0	5.4	0.0	
29	51	1	160	83	96	147	400	301	82	-37	...	0.0	7.3	-3.9	
..	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
422	29	1	162	57	83	164	359	154	69	64	...	0.0	14.1	-2.2	
423	51	0	186	95	94	203	367	171	106	-7	...	0.0	9.6	-3.5	
424	7	0	119	21	140	157	438	226	81	-40	...	0.0	10.0	-2.1	
425	36	0	171	93	87	150	362	177	96	44	...	0.0	10.3	-0.8	
426	35	1	160	53	55	163	340	162	102	40	...	0.0	8.7	-0.5	
427	58	0	160	65	133	148	417	260	92	-158	...	-0.4	6.4	-3.5	
428	64	0	160	63	83	0	364	120	90	29	...	0.0	6.7	-0.4	
429	8	1	130	24	77	125	358	159	70	87	...	0.0	11.3	-2.1	
430	11	0	138	29	123	145	361	221	80	112	...	-3.4	19.6	-4.2	

431	47	0	166	56	79	145	381	173	101	52	...	0.0	8.5	0.0
432	11	0	140	42	88	123	362	228	81	-18	...	0.0	17.1	-7.1
433	70	0	167	60	80	149	290	128	93	-67	...	0.0	2.7	-5.4
434	20	0	178	65	88	155	360	163	71	-22	...	-0.5	10.2	0.0
435	39	1	164	62	79	155	367	153	95	50	...	0.0	9.7	-0.7
436	32	1	164	57	77	144	340	148	82	27	...	-0.6	9.9	-0.6
437	35	1	155	63	87	142	391	137	88	66	...	0.0	10.7	0.0
438	37	0	175	82	88	146	357	179	72	1	...	-0.4	13.5	-1.2
439	49	1	168	66	94	170	383	152	115	92	...	0.0	8.2	-0.7
440	37	0	176	72	88	153	389	172	89	67	...	-0.9	16.6	-3.4
441	37	1	160	50	74	143	374	146	75	68	...	0.0	11.4	-0.9
442	65	1	160	50	85	143	363	146	84	-40	...	0.0	6.6	-6.1
443	41	1	154	75	88	157	384	132	112	65	...	-0.4	10.5	-2.5
444	29	0	166	63	81	143	325	218	74	24	...	0.0	7.8	-1.3
445	45	0	175	75	91	134	376	160	83	91	...	0.0	7.1	-2.4
446	20	1	157	57	81	151	363	166	80	43	...	0.0	7.2	-0.7
447	53	1	160	70	80	199	382	154	117	-37	...	0.0	4.3	-5.0
448	37	0	190	85	100	137	361	201	73	86	...	0.0	15.6	-1.6
449	36	0	166	68	108	176	365	194	116	-85	...	0.0	16.3	-28.6
450	32	1	155	55	93	106	386	218	63	54	...	-0.4	12.0	-0.7
451	78	1	160	70	79	127	364	138	78	28	...	0.0	10.4	-1.8

	273	274	275	276	277	278	279
0	0.0	0.0	0.9	2.9	23.3	49.4	8
1	0.0	0.0	0.2	2.1	20.4	38.8	6
2	0.0	0.0	0.3	3.4	12.3	49.0	10
3	0.0	0.0	0.4	2.6	34.6	61.6	1
4	0.0	0.0	-0.1	3.9	25.4	62.8	7
5	0.0	0.0	0.9	2.2	13.5	31.1	14
6	0.0	0.0	0.4	1.0	14.3	20.5	1
7	0.0	0.0	0.1	0.5	15.8	19.8	1
8	0.0	0.0	0.6	2.1	12.5	30.9	1
9	0.0	0.0	0.8	0.9	20.1	25.1	10
10	0.0	0.0	0.8	0.9	12.3	19.3	3
11	0.0	0.0	0.5	1.5	4.9	17.2	1
12	0.0	0.0	0.8	0.5	8.8	12.1	10
13	0.0	0.0	0.4	2.1	28.5	48.6	6
14	0.0	0.0	0.3	1.7	39.2	54.1	1
15	0.0	0.0	0.6	1.7	17.2	31.1	1
16	0.0	0.0	0.6	2.3	19.5	41.1	10
17	0.0	0.0	0.3	0.7	17.1	20.8	1
18	0.0	0.0	0.4	0.5	18.2	22.4	1
19	0.0	0.0	0.5	1.8	25.2	38.5	1
20	0.0	0.0	0.6	3.3	17.1	54.7	1
21	0.0	0.0	0.8	1.1	11.7	19.6	1
22	0.0	0.0	0.5	1.7	19.7	34.3	1
23	0.0	0.0	0.4	1.1	18.4	28.9	1
24	0.0	0.0	0.5	2.5	35.3	57.3	1

```
25   0.0  0.0  0.4  1.3  20.7   29.2   16
26   0.0  0.0  0.6  2.6  44.0   68.4   14
27   0.0  0.0  0.3  3.2  25.4   54.8   10
28   0.0  0.0  0.4 -1.4  17.2    3.0    2
29   0.0  0.0  0.5 -1.1   3.6   -6.3    2
..   ...  ...  ...  ...   ...    ...  ...
422  0.0  0.0  0.5  3.0  32.7   56.1    1
423  0.0  0.0  1.0  1.6   9.4   23.4    1
424  0.0  0.0  1.0  5.5  36.7  115.9    9
425  0.0  0.0  0.6  3.0  24.1   52.9    1
426  0.0  0.0  0.5  2.3  20.9   40.6    1
427  0.0  0.0  0.4  0.8  -2.9    6.5   10
428  0.0  0.0  0.3  0.4  23.7   26.4    1
429  0.0  0.0  0.7  3.6  16.1   49.2   16
430  0.0  0.0  0.2  1.8  12.2   25.1   10
431  0.0  0.0  0.6  1.2  20.4   29.0    6
432  0.0  0.0  0.7  5.5  15.1   84.4   10
433  0.0  0.0  0.3 -0.2  -7.1   -8.3    3
434  0.0  0.0  0.5  0.4  24.0   25.4    1
435  0.0  0.0  0.8  1.3  24.1   33.7    1
436  0.0  0.0  0.5  2.4  19.1   36.3    1
437  0.0  0.0  1.0  2.1  25.6   43.2    1
438  0.0  0.0  0.5  0.6  30.1   35.0    1
439  0.0  0.0  0.8  1.7  21.5   33.7    1
440  0.0  0.0  0.7  1.8  24.9   41.4    1
441  0.0  0.0  0.7  1.8  40.1   55.5    1
442  0.0  0.0  0.5  0.5  -3.8    0.4    1
443  0.0  0.0  0.5  1.4  17.8   29.5   10
444  0.0  0.0  0.5  2.3  14.1   37.1    1
445  0.0  0.0 -0.4  1.3   8.5   17.6    1
446  0.0  0.0  0.5  2.3  17.6   39.2    1
447  0.0  0.0  0.7  0.6  -4.4   -0.5    1
448  0.0  0.0  0.4  2.4  38.0   62.4   10
449  0.0  0.0  1.5  1.0 -44.2  -33.2    2
450  0.0  0.0  0.5  2.4  25.0   46.6    1
451  0.0  0.0  0.5  1.6  21.3   32.8    1

[452 rows x 280 columns]
```

5.1 Part a

There are 452 patients and 279 features.

5.2 Part b

The first 4 features are age, gender, weight, and height respectively.

5.3 Part c

It's extremely hard to say which values are missing and which are legitimately 0. This data set would be much more easier to prune if missing values were represented as `None`.

5.4 Part d

I would do a decision tree betwen each feature and the condition and then measure the error of the decision stump. If the error is small, that feature is clearly a strong influencer for the patient's condition.

6 Written Number 2

```
In [22]: A=np.matrix('1 1; 1 0; 1 0')
         A

Out[22]: matrix([[1, 1],
                 [1, 0],
                 [1, 0]])

In [23]: def calc_projection(A, point):
             A_T = np.transpose(A)
             A_T_times_A_inverted = np.linalg.inv(np.matmul(A_T, A))
             matrices_multiplied = np.matmul(np.matmul(A, A_T_times_A_inverted), A_T)
             return np.matmul(matrices_multiplied, point)

In [24]: p_1_proj = calc_projection(A, np.array([3,3,3]))
         p_1_proj

Out[24]: matrix([[3., 3., 3.]])

In [25]: p_2_proj = calc_projection(A, np.array([1,2,3]))
         p_2_proj

Out[25]: matrix([[1. , 2.5, 2.5]])

In [26]: calc_projection(A, np.array([0,0,1]))

Out[26]: matrix([[1.11022302e-16, 5.00000000e-01, 5.00000000e-01]])
```

7 Written Number 3

```
In [2]: from scipy.stats import binom

        binom.cdf(50, 100, 2/3) * 100

Out[2]: 0.04193410826744501

In [ ]:
```